REMARKS

The applicants have carefully considered the Office action dated September 24, 2007 and the references it cites. By way of this Response, claims 1-6, 9-16, 21, 24-28, 30, 32 and 36 have been amended, claim 35 has been cancelled without prejudice to its further prosecution and new claim 37 has been added. In view of the foregoing amendments and following remarks, the applicants respectfully submit that all pending claims are in condition for allowance and early notice to that effect is respectfully requested.

35 U.S.C. § 101 Rejections

The Office action rejected claims 25-34 as directed to non-statutory subject matter under 35 U.S.C. § 101. The Office action indicated that amending the preamble to include the phrase "having a processor" would overcome this rejection. Applicants have followed this suggestion with a broader formulation, namely, the applicants have added the phrase "having a logic circuit" to the preamble of claim 25 to make it clear that the claim encompasses apparatus including either a processor or other hardware in at least one of the elements. Support for this amendment can be found throughout the specification including, for example, at paragraphs [0043] and [0089] – [0094]. Accordingly, withdrawal of the rejections under 35 U.S.C. § 101 is respectfully requested.

Art Rejections

Turning to the art rejections, the Office action rejected claims 1-10 and 21-36 under 35 U.S.C. § 102(e) as being anticipated by Mahadevan et al. (U.S. Patent No. 5,797,013) and rejected claims 11-20 under 35 U.S.C. § 103(a) as being obvious over Mahadevan et al. in

view of Granston et al. (U.S. Patent Publication No. 2003/0140334, now U.S. Patent No. 7,237,234).. The applicants respectfully traverse these rejections.

By way of background, the applicants note that the instant application discloses methods and apparatus to dynamically optimize a program undergoing binary translation. The methods and apparatus implement a multiple stage optimization process. In a first stage, referred to as cold translation, the methods and apparatus translate the program (e.g., instructions adapted for execution in a first instruction set architecture such as an Intel x86 processor) into native language (e.g., a second instruction set format) to quickly enable the entire program to execute on the platform in question (e.g., an Itanium® processor). This first process results in a cold translated program that is executable on the target platform, but is not necessarily optimized.

Thus, the methods and apparatus of the then enters a hot translation phase in which one or more blocks of the cold translated program are identified as possible candidates for optimization and instrumented to enable a determination as to whether they are used frequently enough to justify optimization. The hot translated blocks are linked into the cold translated program so that the cold translated program and hot translated blocks can continue to be executed and information collected to enable identification of one or more hot loops. When one or more hot loops are identified, a third phase is initiated, (referred to herein as a gen-translation), in which instrumentation is inserted into the hot loop in order to collect profile data for one or more load instructions. The instrumented hot loop is linked into the cold translated program to enable the instrumented hot loop to execute in the execution path of the cold translated program.

The methods and apparatus then enter a fourth phase in which one or more prefetching instructions are inserted into the hot loop to prefetch data for one or more load

instructions. The hot loop with the prefetching instruction(s) is then linked into the cold translated program such that the cold translated program executes with the optimized hot loops. One or more other blocks of instructions may then be executed.

Advantageously, the example methods and apparatus enable quick translation and, thus, executability of the program and then, subsequently, optimize one or more blocks of the program in series or parallel to improve execution over time. An advantage of this approach is that the program is quickly usable and portion(s) of the program that would benefit from optimization are identified over time based on actual execution patterns to ensure the block(s) of the program is optimized in an order and approach that makes sense.

The claims have been amended to better reflect the above multiple phase approaches. For example, claim 1 now recites a method to optimize a program comprising, among other things, cold translating a plurality of blocks of the program from a first language to a second language to generate a cold translated program; determining a cold execution trip count associated with a first one of the blocks in the cold translated program; identifying the first block for hot translation when the cold execution trip count exceeds a first threshold; hot translating the first block into a first hot translated block; linking the first hot translated block into the cold translated program; identifying the first hot translated block as associated with a hot loop when the hot execution trip count exceeds a second threshold; inserting instrumentation into the hot loop to develop profile data for a load instruction within the instrumented hot loop; linking the instrumented hot translated block into the cold translated program; and inserting a prefetching instruction into the hot loop if the profile data indicates the load instruction in the instrumented hot loop meets a predefined criteria. No art of record, taken alone or in combination, teaches or suggests such a method.

For example, Mahadevan et al. is directed to a method to unroll loops in a compiler, which converts source code in a programming language into machine readable instructions. Errored branch predictions incur significant performance penalties. Thus, to reduce the number of branch predictions that occur (and, thus, the opportunities for mis-prediction), the contents of the loop (i.e., the instructions inside the loop) are duplicated, or unrolled, by the unroll factor. However, none of the Mahadevan disclosure has anything to do with the multiple phase process of claim 1. More specifically, Mahadevan has no teaching or suggestion of cold translating a program, hot translating a block of the program, linking the hot translated program into the cold translated program, identifying a hot loop, instrumenting the hot loop, or linking the instrumented hot loop into the cold translated program as recited in claim 1.

Granston, which relates to soliciting user input during a compiling process, is also irrelevant to claim 1. Thus, irrespective of how one combines Mahadevan and Granston, one does not arrive at the method of claim 1. Accordingly, claim 1 and all claims depending therefrom are in condition for allowance.

Independent claim 5 is also allowable. Claim 5 recites a method comprising, among other things, cold translating a program from a first instruction set to a second instruction set to generate a cold translated program; hot translating at least one block of the cold translated program to generate at least one hot translated block; linking the at least one hot translated block into the cold translated program to generate a hot translated program; identifying a hot loop associated with the hot translated block; gen-translating the hot loop to insert instrumentation into the hot loop to attempt to identify a load which would benefit from prefetching; and, if the hot loop identifies the load, use-translating the hot loop to insert a prefetch instruction associated with the load. Neither Mahadevan nor Granston has any

disclosure of cold translating, hot translating, gen-translating or use-translating as recited in claim 5. Therefore, independent claim 5 and all claims dependent therefrom are in condition for allowance.

Independent claim 25 is also allowable. Claim 25 recites an apparatus to optimize a program comprising, among other things, a cold translator to translate the program from a first instruction set to a second instruction set to generate a cold translated program; a hot translation module to hot translate at least one block in the cold translated program; a code linker to link the at least one hot translated block into the cold translated program to generate a hot translated program; a hot loop identifier to identify a hot loop in the hot translated program and to determine if the hot loop should be gen-translated; a gen-translator to instrument the hot loop with instructions to collect profile information; and a use-translator to insert a prefetch instruction associated with the hot loop based on the profile information. Neither Mahadevan nor Granston has any disclosure of a cold translator, a hot translator, a gen-translator or a use-translator as recited in claim 25. Therefore, independent claim 25 and all claims dependent therefrom are in condition for allowance.

Independent claim 37 also. Claim 37 recites machine readable medium storing instructions which are, among other things, structured to cause a machine to: cold translate a program from a first language to a second language to generate a cold translated program; determine a cold execution trip count associated with a first one of the blocks of the cold translated program; identify the first block for hot translation when the cold execution trip count exceeds a first threshold; when the first block is identified for hot translation, hot translate the first block into a first hot translated block; link the hot translated block into the cold translated program to generate a hot translated program; identify the first hot translated block as associated with a hot loop when the hot execution trip count exceeds a second

threshold; insert instrumentation into the hot loop to develop profile data for a load instruction within the instrumented hot loop; link the instrumented hot loop into the hot translated program; and insert a prefetching instruction into the hot loop if the profile data indicates the load instruction in the instrumented hot loop meets a criteria. Neither Mahadevan nor Granston, whether taken alone or in combination, teaches or suggests such machine readable instructions. Therefore, independent claim 37 and all claims dependent thereon are in condition for allowance.

In view of the foregoing, the applicants respectfully submit that all pending claims are in condition for allowance. If there is any matter that the examiner would like to discuss, the examiner is invited to contact the undersigned representative at the telephone number set forth below.

Respectfully submitted,

HANLEY, FLIGHT & ZIMMERMAN, LLC
150 S. Wacker Dr.
Suite 2100
Chicago, Illinois 60606

Dated: **March 24, 2007**

/James A Flight/

James A Flight
Reg. No. 37,622
Attorney for Applicants
312.580.1020